Entrega Tema 3 – Base de Datos

Diseño Físico de Base de Datos

“Agencia de Viajes”

Desarrollo de Aplicaciones Web (DAW)

Base de Datos

Cesur

Curso 1º

2023-2024

Andy López Rey

10/03/2024

Modelo Entidad - Relación Extendido
y
Modelo Relacional

Generalización / Especialización

En este caso el Modelo Entidad – Relación (E-R) Extendido no posee ningún tipo de generalización/especialización.

Modelo E – R Extendido

Pasaremos a explicar detalladamente las entidades, relaciones y cardinalidades del Modelo E-R Extendido. Las relaciones están representadas por una flecha (→).

- Sucursal → Turista. Relación 1:N y 1:M. Cardinalidad máxima N:M.

  Nos comentan que interesa especialmente conocer qué sucursal ha contratado el turista. Esto lo podemos representar con una única relación entre Sucursal y Turista. La cadena de agencias sabemos que está compuesta por todo un conjunto de sucursales del que el turista podrá seleccionar una o varias, en caso de que quiera (1:M). Por otro lado las sucursales tendrán un gran número de turistas a su disposición para ofrecer sus servicios (1:N).

  De acuerdo a las reglas de Transformación del Modelo Relacional, al tener una cardinalidad máxima N:M esta relación generará una nueva relación (o tabla) que en nuestro caso hemos llamado TuristaContrataSucursales.

  También cabe comentar que los atributos multivaloradores (teléfonos de sucursales y turistas) generan también una relación cada uno.

  Se ha optado por no tener una relación 0:N porque se presupone que si un turista no quiere contratar una sucursal de la agencia de viajes no interesan sus datos.

- Turista → Vuelo. Relación 1:N y N:M. Cardinalidad máxima N:M.

  La agencia de viajes tiene toda una serie de vuelos en contrato que ofrecen a los turistas que contratan las sucursales. Los turistas pueden optar por comprar un vuelo o muchos (1:N). Recordemos que en ocasiones es necesario tomar varios vuelos para una misma ruta por escalas etc. Los vuelos tendrán un gran número de turistas dentro. No se realiza una relación con cardinalidad 1:N (sino N:M) porque al tratarse de una cadena de viajes para turistas se asume que no hay vuelos privados que se puedan contratar. Y por tanto, una situación donde un vuelo tenga a una única persona como turista se ha descartado.

  Se requiere que es interesante saber en qué clase volarán los turistas así como el número de plazas que estos compran. La clase, que será primera o en turista, nos vendrá dada por el atributo de la relación Vuela. Las plazas sin embargo se ha optado por no ponerlas como atributo en la relación (que también se podría) porque es posible saberlo con los atributos que ya están presentes en la entidad Vuelo. Conociendo en qué clase viajará el turista, sabremos qué tipos de plaza se han de descontar del total del vuelo. Recordemos que al sólo haber clases en primera y turista, con una simple operación matemática se podría saber qué

cantidad de plazas se han adquirido y hay disponibles. Sería algo que se solucionaría a nivel de programación.

Al igual que antes se generará una nueva relación llamada TuristaTomaVuelos que nos indicará los vuelos que los turistas han contratado y en qué clase viajan.

- Turista → Hotel. Relación 1:N y 1:M. Cardinalidad máxima N:M.

  La agencia ofrece todo un repertorio de hoteles en los que el turista se podrá hospedar. Un turista podrá hospedarse en 1 o varios hoteles (1:N). Se ha hecho con una relación a varios porque es muy posible que los turistas contraten varias hoteles durante un mismo viaje en caso de que sea por varios días o se muevan por el territorio del destino. En este caso se ha optado por una relación 1:M de Hotel → Turista y no N:M (como en los vuelos) porque los turistas pueden viajar solos. Si viajan sin acompañantes entonces posibilita la relación a uno.

  Nos requieren que es importante saber el régimen de hospedaje que el turista ha contratado, número de habitaciones reservadas y la fecha de Check-In y Check-Out. Toda esta información la podremos saber gracias a los atributos de la relación.

  Al igual que con vuelos se podría omitir el atributo de habitaciones reservadas para conocer el número de habitaciones que el turista compra si tomamos los datos del régimen comprado. Sin embargo a diferencia de antes sería algo muy rebuscado a nivel de programación y fácil de solucionar a nivel de base de datos.

  La relación origina una tabla denominada TuristaHospedaHoteles que contendrá los atributos mencionados previamente.

Aclaraciones / Restricciones

Base de Datos Física

Tenemos una serie de atributos que requerirán de un tipo de datos ENUM a la hora de construir la base de datos física.

- Clase → Turista, Primera.
- Régimen de Hospedaje → Desayuno, Media Pensión, Pensión Completa.

Otro atributo interesante es la Fecha y Hora de la entidad Vuelo. Tendrá que ser un tipo de dato DATETIME y no DATE, pues se requiere saber la hora también.

Aunque en la entidad Hotel no se pide una hora de llegada/salida, se optará por modelarla como un DATETIME también. Esto es debido a que hay podría haber casos en los que un turista haga Check-In y Check-Out el mismo día. Y con un tipo de dato DATE podría generar problemas.

Restricciones

- Como se comentó antes se ha dejado una restricción a propósito en la entidad Vuelo. Aunque podríamos añadir un atributo a la relación para saber cuántos asientos un turista adquiere, con ya saber en qué clase van es algo que se puede inferir pero que se tendrá que programar.
- Sucursal sólo se ha relacionado con Turista. Se ha optado por modelarlo así porque se presupone que un turista sólo podrá contratar sucursales que estén dentro del conjunto que la

agencia de viajes dispone y estas disponen ya contratos de exclusividad con los hoteles y vuelos. Así la información de los vuelos y hoteles se podrán saber.

Pero esto quiere decir que se tendrá que restringir de alguna forma que los turistas sólo puedan contratar dichas sucursales y preservar así la integridad referencial.

Modelo E – R Extendido → Modelo Relacional

El modelo relacional es el resultado de aplicar las reglas de la transformación sobre un modelo E-R extendido y obtener toda una serie de relaciones o tablas.

Claves primarias / Primary Keys: subrayado.
Claves foráneas / Foreign Keys: subrayadas y con asterisco (*).

Resultado final después de realizar la Transformación al Modelo Entidad – Relación Extendido:

Sucursales (ID_Sucursal, Dirección)
Teléfonos_Sucursal (Sucursal_ID*, Teléfono_Sucursal)
Turistas (ID_Turista, Nombre_Pila, Apellido_1, Apellido_2, Dirección)
Teléfonos_Turista (Turista_ID*, Teléfono_Turista)
TuristaContrataSucursales (Turista_ID*, Sucursal_ID*)
Vuelos (Nº_Vuelo, Origen, Destino, Fecha_y_Hora, Plazas_Totales, Plazas_Turista)
TuristaTomaVuelos (Turista_ID*, Vuelo_ID*, Clase)
Hoteles (ID_Hotel, Nombre, Categoría, Dirección, Ciudad, Nº_Plazas_Disponibles)
Teléfonos_Hotel (Hotel_ID*, Teléfono_Hotel)
TuristaHospedaHoteles (Turista_ID*, Hotel_ID*, Fecha_Llegada, Fecha_Partida, Régimen_H, Nº_Habs_Reservada)

Normalización (FN1 - FN3)

Forma Normal 1 (FN1)

En la primera fase normal o FN1 se prohíbe que en una tabla haya atributos que puedan tomar más de un valor a la vez.

Aquellos atributos que puedan tomar más de un valor a la vez, deberán de ser eliminados de la tabla y formarán una entidad completamente nueva por su propia cuenta.

Al momento de construir un modelo relacional, por la forma de construcción de este tipo de gestor ya hace que no sea posible almacenar más de un valor en el mismo atributo.

Por ende podemos decir que el modelo sí cumple la FN1.

Forma Normal 2 (FN2)

Para aplicar la segunda forma normal o FN2 el modelo ha de estar en FN1. Hemos comprobado anteriormente que efectivamente lo está.

La FN2 dicta que cada atributo que no forma parte de la clave principal tiene que tener una dependencia funcional completa de ella.

Una vez se identifican los atributos que *no* dependan da la clave principal, se formará con ellos una

nueva entidad o relación y se eliminarán de la original. Además, la clave primaria de la nueva entidad estará formada por parte de la antigua de la que sí dependían funcionalmente los atributos (en caso de ser compuestas).

Así pues analicemos relación por relación.

- **Sucursales**. Contiene la información del repertorio de sucursales pertenecientes a la agencia de viajes. Sólo contiene un atributo que no forma parte de la clave principal. Podemos decir que Dirección depende funcionalmente de ID_Sucursal porque cada sucursal tendrá una dirección específica.
- **Teléfonos_Sucursal.** Se trata de una relación originada a partir de la transformación de un atributo multivalorado. Sólo tiene su clave principal y la foránea; la cual no se analiza para la normalización. Por tanto no es necesario comprobar si está normalizada o no.
- **Turistas**. Posee la información de todos los turistas que han contratado en algún momento una o varias de las sucursales que posee la agencia de viajes. De la clave principal, ID_Turista, cuelgan atributos personales como el nombre de pila, primer y segundo apellido y la dirección. Todos se tratan de atributos personales que sin una identificación de la persona no se podrían asignar a un individuo único. Por ende poseen una dependencia funcional completa.
- **Teléfonos_Turista**. Caso similar al de Teléfonos_Sucursal.
- **TuristaContrataSucursales**. Se trata de una tabla o relación generada por la relación N:M entre Turista y Sucursal. Gracias a ella podremos saber la información sobre qué sucursal(es) ha contratado el turista. Sólo posee las dos claves foráneas de las entidades de las que se generó. No es necesario analizar si está normalizada.
- **Vuelos**. La entidad Vuelo posee toda la información relacionada con los vuelos sobre los que las sucursales tienen contrato de exclusividad. De la clave principal Nº_Vuelo cuelgan: origen, destino, fecha_y_hora, plazas_totales y plazas_turistas. Para saber el origen y el destino de un vuelo es necesario obviamente saber el vuelo del que se habla. Hay muchos vuelos que salen de un aeropuerto y tienen el mismo destino y origen. La única forma de diferenciarlos es mediante la clave principal. La fecha_y_hora también. Es imposible que varios vuelos salgan a la vez. Hay un control estricto sobre qué vuelos aterrizan y cuáles despegan para evitar desastres y organizar el espacio en la pista y aéreo. Finalmente las plazas totales y turistas también necesitan de un identificador único para poder tener sentido. Podemos decir por tanto que todos los atributos efectivamente poseen una dependencia funcional completa de Nº_Vuelo.
- **TuristaTomaVuelos**. Relación generada a partir de la cardinalidad N:M de Turista → Vuelo. Contiene la información necesaria para saber qué vuelo(s) toman los turistas así como la clase. Tenemos una clave compuesta en esta ocasión: Turista_ID* y Vuelo_ID* y como clave principal Clase. Al no haber atributos no clave no es necesario normalizar.
- **Hoteles**. La entidad Hotel almacena la información que interesa sobre los hoteles que las sucursales poseen a su disposición para ofrecer a los turistas. De la clave principal ID_Hotel cuelgan los atributos: nombre, categoría, dirección, ciudad y nº_plazas_disponibles. Podemos ver claramente que todos los atributos dependerán completamente de hotel. Son todos atributos que para tener sentido y una única existencia han de colgar de un código de hotel. Están definidas unívocamente por ID_Hotel.
- **Teléfonos_Hotel**. Al igual que las otras entidades originadas a partir de un atributo multivalorado no es necesario normalizar.
- **TuristaHospedaHoteles**. Por último tenemos la relación originada de la relación N:M de Turista → Hotel. Como clave compuesta heredada de las entidades de las que cuelga tiene Turista_ID* y Hotel_ID* y como clave principal propia Fecha_Llegada. Es necesario que la fecha de llegada (y hora como se ha explicado antes) sea principal para tener instancias únicas en las que se hace Check-In. Los atributos tendrán que depender completamente de

las claves mencionadas. Fecha_Partida depende unívocamente de las claves porque sin un hotel concreto, el turista y un momento exacto del que se haga Check-In no sería posible definirla. Lo mismo ocurre con el régimen y el número de habitaciones reservadas. Cada persona comprará un régimen y una habitación concreta en un hotel determinado.

Podemos concluir que el modelo relacional presentado anteriormente sí cumple la FN2 ya que todos los atributos elegibles tienen dependencia funcional completa de la clave principal y/o clave compuesta.

Forma Normal 3 (FN3)

La tercera y última fase de normalización o FN3 se da cuando los elementos están en FN2 y además todos los atributos no principales dependen directamente de la clave principal.

Es decir, no deben de haber atributos no principales que dependan de forma transitiva de la clave principal.

De esta forma la FN3 asegura que todos los atributos no clave dependan directamente de la clave y, en ningún caso, dependa de ella de forma indirecta. Por tanto los atributos no claves no pueden pasar por un tercero para así entonces poder depender de la clave primaria.

La definición "formal" sería: A → (determina un único) B → (Determina un único) C, pero B (NO determina un único) → A.

- **Sucursales** (ID_Sucursal, Dirección)

  Hay un único atributo que no forma parte de la clave principal.

  ID_Sucursal → Dirección. No hay una tercera instancia para comprobar. Al cumplir la FN2 basta.

- **Teléfonos_Sucursal** (Sucursal_ID*, Teléfono_Sucursal)

  No hay atributos a analizar.

- **Turistas** (ID_Turista, Nombre_Pila, Apellido_1, Apellido_2, Dirección

  Aclaración: Nombre_Pila, Apellido_1 y Apellido_2 son resultado de la transformación de un atributo compuesto. Pero para nuestro análisis tomaremos estos tres como un único atributo: Nombre. En caso contrario se rompería la FN3, pues un único Nombre_Pila no determina un DNI y Apellido; ya que pueden repetirse muchos nombres, apellidos etc. Y se tendría que crear una nueva tabla sólo con los nombres.

  ID_Turista → Nombre. Veremos si Nombre → algún atributo no clave:

  Nombre → Dirección. Cada persona tiene una dirección asignada a ella. Y Nombre → ID_Turista. Cada nombre completo nos dará el código del turista. Por tanto no hay dependencias transitivas.

  De igual manera Dirección NO → Nombre. Necesita del código del turista que determina la persona.

- **Teléfonos_Turista** (Turista_ID*, Teléfono_Turista)

  No hay atributos a analizar.

- **TuristaContrataSucursales** (Turista_ID*, Sucursal_ID*)

  Se trata de una clave compuesta heredada de las entidades. No hay atributos a analizar.

- **Vuelos** (Nº_Vuelo, Origen, Destino, Fecha_y_Hora, Plazas_Totales, Plazas_Turista)

  Nº_Vuelo → Origen. Veamos si Origen → algún atributo no clave.

  Origen → (Destino, Fecha_y_Hora, Plazas_Totales, Plazas_Turista). No hay ningún atributo que se pueda determinar únicamente a través de origen, es decir, que tenga una dependencia transitiva de Origen. Es necesario la clave principal.

  Nº_Vuelo → Destino. Veamos si Destino → algún atributo no clave.

  Destino → (Origen, Fecha_y_Hora, Plazas_Totales, Plazas_Turista). Nuevamente tampoco es posible determinar el destino únicamente a través de esos atributos. Los vuelos pueden compartir origen y plazas etc. Por tanto necesitarán del vuelo en concreto al que nos referimos; de la clave principal.

  Nº_Vuelo → Fecha_y_Hora. Fecha_y_Hora → atributos no clave. A través de la fecha y hora únicamente no es posible determinar unívocamente algún atributo no clave.

  Nº_Vuelo → Plazas_Totales y Plazas_Turista. Obviamente tampoco habrá dependencia transitiva. Sabiendo únicamente las plazas no es posible saber el destino u origen de un vuelo.

- **TuristaTomaVuelos** (Turista_ID*, Vuelo_ID*, Clase)

  Tenemos la clave principal de la relación y la compuesta heredada. No es necesario verificar si cumple la FN3 al no haber atributos no clave. Por definición ya está en FN3.

- **Hoteles** (ID_Hotel, Nombre, Categoría, Dirección, Ciudad, Nº_Plazas_Disponibles)

  El razonamiento en este caso es muy similar a la entidad Vuelo.

  ID_Hotel → Nombre, pero sólo con el nombre no Nombre → algún atributo no clave.

  ID_Hotel → Categoría. Pero Categoría NO → algún atributo no clave. Sabiendo la categoría no podemos determinar ningún otro atributo.

  ID_Hotel → (Dirección, Ciudad). Pero (Dirección, Ciudad) → NO algún atributo no clave. No hay dependencia transitiva pues no se puede definir un hotel unívocamente sólo con la dirección y la ciudad.

  ID_Hotel → Nº_Plazas_Disponibles. Claramente Nº_Plazas_Disponibles → NO atributos no clave.

- **Teléfonos_Hotel** (Hotel_ID*, Teléfono_Hotel)

  No hay atributos a analizar.

- **TuristaHospedaHoteles** (Turista_ID*, Hotel_ID*, Fecha_Llegada, Fecha_Partida, Régimen_H, Nº_Habs_Reservada)

  Finalmente tenemos un caso con una clave compuesta.

  Fecha_Partida NO determinará → Régimen_H y Nº_Hab_Reservada sin la dependencia completa sobre la clave compuesta y la principal.

  Régimen_H NO → Fecha_Partida, Nº_Habs_Reservada. Sólo sabiendo el régimen que el turista adquirió es imposible saber algo del resto de atributos. Es algo que se comparte entre muchas personas en un hotel.

  Y Nº_Habs_Reservadas NO → algún atributo no clave. Es imposible definir unívocamente otro atributo sólo sabiendo cuántas habitaciones ha reservado.

Al analizar cada atributo no clave y ver que ninguno depende transitivamente de la clave principal y/o compuesta podemos decir que el Modelo Relacional aquí presente sí que cumple con la FN3.

Nos garantizará una cierta calidad a nuestro modelo y evitará redundancias y una mayor eficiencias a la hora de realizar la base de datos a nivel físico.